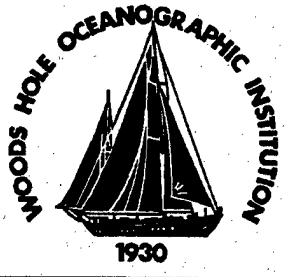
WHOI-98-04

Woods Hole Oceanographic Institution



A Graphical User Interface for Processing Data from the High Resolution Profiler (HRP)

by

Ellyn T. Montgomery and S. Thompson Bolmer

March 1998

Technical Report

Funding was provided by the National Science Foundation through Grant No. OCE-94-15589.

Approved for public release; distribution unlimited.

WHOI-98-04

A Graphical User Interface for Processing Data from the High Resolution Profiler (HRP)

by

Ellyn T. Montgomery and S. Thompson Bolmer

Woods Hole Oceanographic Institution
Woods Hole, Massachusetts 02543

March 1998

Technical Report

Funding was provided by the National Science Foundation through Grant No. OCE-94-15589.

Approved for Distribution:

Philip L. Richardson, Chair
Department of Physical Oceanography

Table of Contents

List of figures

Appendices–

Abstract–

The goal of observational oceanography is to learn about how the ocean works from data acquired at sea. Many instrument systems are used to collect data that describe ocean processes, and each has its own software tools to access and analyze the data. Some instruments are widely used and commercially available, while others are one of a kind systems, designed and developed to perform a specialized function.

The High Resolution Profiler (HRP) is one of the latter type of oceanographic instrument. It is one of a very few that can collect data for studies of very small scale ocean mixing (turbulence and dissipation). To examine and work with data from the HRP, a unique set of communications, conversion and analysis programs is employed. A major overhaul of this software was completed recently, as described in this document. The considerations and constraints that influenced the software conversion plan and implementation will be discussed in the next section.

To understand the data processing system, one must first know a little bit about the instrument and the kinds of data it collects and stores. For this reason, the HRP, the sequence of events that occur when working at sea, and the raw data format are described in sections following the conversion overview. Following that, the actual programs used and the processing sequence are described. Next, the shell scripts that were developed to automate program execution are described. Finally, the major enhancement, a Graphical User Interface (GUI) that was developed on the Brazil Basin Tracer Release Experiment cruises in 1996 and 1997, is described. Appendices contain much of the actual code used in the processing system.

Software Conversion Overview–

Ten years ago, oceanographers were happy to return from a research cruise with raw data from their instruments archived for processing on land. Now, with new powerful computers, we expect to process and analyze data while at sea. This is a significant advance, because now, data collected early in a cruise can be used to determine the sampling strategy later on the same cruise.

The software described in this report is used to process the data collected by the High Resolution Profiler (HRP). Regardless of the computer used to do the processing, a series of communication and computation steps must take place in order for the information acquired by the HRP during a profile to be useful for subsequent planning or analysis. The routine that normally occurs after each HRP dive is the following:

- three types of data are transferred from the HRP to a PC
- the files are transferred to a workstation for processing
- the raw fine and microstructure data are plotted for quality control purposes
- sensor calibrations are applied, velocities are computed, and half decibar averaged profiles of the following properties are stored:
  - pressure
  - temperature
  - salinity
  - north velocity
  - east velocity
- a plot of the corrected fine scale data is generated
- diffusivity estimates are computed using the microstructure data
- histogram plots of diffusivity data are made
- the fine and microstructure data are combined and stored as ASCII files

Both the HRP and its software were originally developed at the Woods Hole Oceanographic Institution in the late 1980's. Fortran77 programs were written to operate on Digital Equipment Corporation (DEC) VAX/VMS computers. At the time, this hardware was the cutting edge of mainframe and desktop computing. In recent years maintaining the DEC microvax computers became prohibitively expensive, and they were slow compared to newer computers. New hardware was required, and a detailed evaluation of the possibilities was made. In 1993, SGI UNIX workstations were selected and purchased to replace the microvaxes.

Choosing a UNIX platform necessitated converting all the existing software for use on the new computers. Since a major effort was required to port the existing software to UNIX, we decided to enhance the system at the same time. The goal was to develop an efficient, easy–to–use method of processing the data from the HRP. Four main considerations dominated the planning for the change of systems: future portability, efficiency of the process, accuracy of the computations, and ease of use. The issues associated with each are described briefly in the

following paragraphs.

With the rapid changes in the computer industry, it is likely that new computers will be purchased more often than in the past. For this reason, making the software portable during this conversion was desirable. The new processing system will not contain any platform–specific code to avoid future potential incompatibilities.

The first porting issue is that the VAX and SGI computers store their data in opposite byte orders. The VAX, PC, and HRP are all "little endian" (least significant byte is written first), the SGI is "big endian" (most significant byte written first). Consequently, binary data from one system cannot be read directly on the other. Fortunately, the UNIX "dd" command has an option for swapping bytes, so prior to use, all raw data transferred to the SGIs gets byte–swapped using this command :

"**dd if=fine###.raw of=fine###_s.raw conv=swab**".

The shell scripts that control the first use of raw data issue this command, and then the byte swapped file is used as input to everything else. Note the byte–swapped output file has "**_s**" appended to the name.

The second porting issue was using "generic" FORTRAN language that would compile and run on any machine. Before compilation on the SGI, any VMS specific code in the programs was replaced with generic language. There also is a very important difference between the compilers: the VAX one assumes that variables retain their values in subroutines, the SGI one does not. To compensate for this, "**–static**" was added to the compiler option flags in the makefile for each program.

The third portability issue was how to output plots and figures. The old plotting software could use graphics libraries and device drivers that were created for use on the old Woods Hole Oceanographic Institution VAX mainframe. The main library was called "whoilib" and only supported antiquated output devices; it would not run on UNIX computers. Converting the plotting software to output PostScript files, which are widely used, easily viewed and output with existing UNIX tools, was the best option. The old programs were modified to incorporate the postscript libraries provided with the GMT program (Generic Mapping Tools, Wessel and Smith, 1991, 1995) instead of the calls to the old whoilib.

No attempts were made to change the FORTRAN 77 code to C, C++ or Perl. The scientists who use and modify the processing programs only know FORTRAN, so language conversion was not feasible. Functional equivalents of some of the programs have been written for The Mathworks Matlab software. These programs are not currently used in the at–sea processing system, but may be in the future.

To optimize the processing system data flow, computational steps were combined, and non–essential intermediate data products removed. There was one instance where this was possible: the .vax format data and associated conversion program pro2ctd_x, were eliminated in the new version of the software. The new programs all work from the byte–swapped raw data where the old ones would have used .vax format inputs. The .vel output from the original velocity computation program also used .vax format. Now on the SGIs, the velocity data is stored as a plain binary data file with the .uvel suffix.

Transferring the data directly from the HRP to a workstation would have improved the efficiency of the overall process. Unfortunately, the HRP only knows how to send a set number of bytes from a memory address requested, so special communications software must be written for the other computer to control the data transfer. The tools for programming the SGIs serial port do not exist, so the program written for the PC must remain in use. The added step of FTPing the data from the PC to the workstation is fast, so does not constrain the processing time significantly.

The results generated by the VAX programs were thoroughly verified during their development, so the VAX generated results were used to verify the results produced by the SGIs. The process was arduous, because the outputs were often binary format, so programs had to be written to convert and compare the results. Most of the programs produced output that matched exactly the VAX output. The **mfft_g** program (in the microstructure processing) however, generates slightly different results on the two platforms. The **scales()** subroutine operates on very small floating point numbers, and when run on the SGIs, returns numbers very slightly different from those generated on the VAXs. These small differences are then propagated through the subsequent computations. To test the size of the error, RMS calculations were made on the final dissipation estimates, and the difference between the means calculated by SGI and VAX is on order 5e–12 for chi and 1e–11 for epsilon. The variances are almost identical between VAX and SGI and vary between 1e–15 and 1e–18 for chi and epsilon respectively. As a result of these tests, we became confident that the data processed on the SGIs is similar enough to that processed on the VAXs that there would be no effect on the conclusions derived from the data.

The main enhancement for ease of use was developing a Graphical User Interface (GUI) for the processing software. The original processing system was cumbersome and involved many labor–intensive steps. VMS Digital Command Language (DCL) instructions were used to provide inputs to the FORTRAN programs and automate some of the processing, but the files all had to be edited by hand. The VMS command files were replaced by a suite of Bourne shell scripts, which use input files that can be modified using sed and awk commands prior to execution. For portability, only generic UNIX commands were used in writing the scripts. Overall process control programs were created to manage running a related series of processes, e.g., all the microstructure computations at once. The GUI tool was designed to interface with the shell scripts by using buttons on pull–down menus to select among various processing

options. Clicking on a button invokes the shell script(s) associated with it to accomplish the desired processing task. Associated with the implementation of the GUI, a method of providing input to the programs was devised that simplifies documentation of the parameters used in processing. It is used by the GUI, and so makes reprocessing significantly easier.

Description of the High Resolution Profiler (HRP)

The following is a very brief description of the High Resolution Profiler (HRP) and a summary of how profiles are made during a research cruise. The development and history of the HRP are described in papers by Schmitt et al, (1986, 1995). The operational details of communicating with and using the HRP are documented in a technical report by Montgomery (1991).

The HRP is a one–of–a–kind oceanographic instrument, designed to measure mixing on very small vertical scales (centimeter to ten meter scales). To obtain measurements that are not contaminated by the ship's motion, the HRP operates without attachment to the ship. It is controlled by its own internal computer and has 16 MB of internal RAM for data storage. The HRP is deployed, falls at a nominal rate of 0.6 m/s while collecting data, stops data acquisition at a user specified pressure or range threshold, puts the computers into a dormant state to conserve power, releases the descent weights, and ascends to the ocean surface where it can be recovered. A schematic of the HRP and its component systems is shown in figure 1.

The HRP has two profiling modes for data acquisition: fine and micro, with the transition between them triggered by the CTD's pressure reaching a user defined threshold value. Fine scale sensors are sampled at 10 Hz, and microstructure sensors are sampled at 200 Hz, with fine sampling continuing throughout the period of micro sampling. All the sensors except those of the CTD are interfaced to the HRP's computer through the A/D converter circuitry. The current sensor configuration and channel assignments are listed below.

fine sensors–	A/D channel
pressure	–
temperature	–
conductivity	–
accelerometer top X	0
accelerometer top Y	1
accelerometer bottom X	2
accelerometer bottom Y	3
acoustic current meter X velocity	4
acoustic current meter Y velocity	5
X magnetometer	6
Y magnetometer	7
A/D ground	14

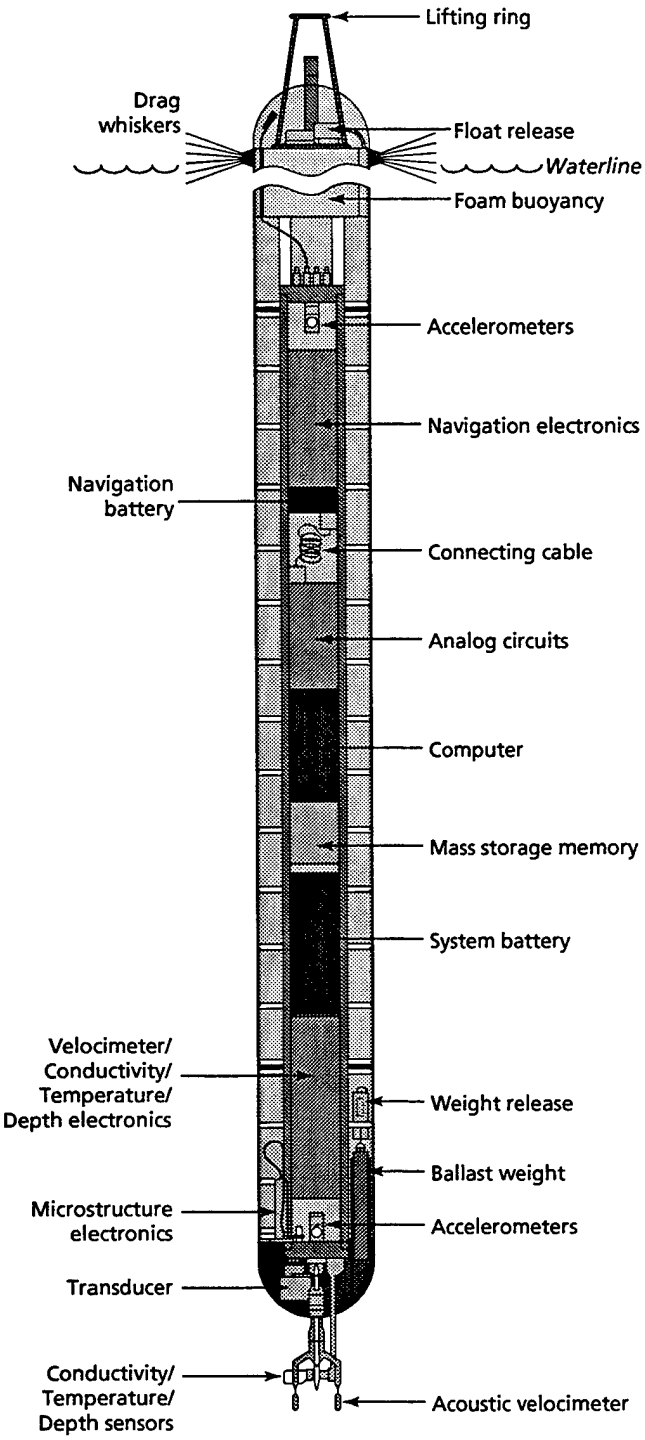


Figure 1 : Schematic Diagram of the High Resolution Profiler (HRP) and its component systems.

micro sensors–	A/D channel
differential conductivity	10
differential temperature	11
shear X	12
shear Y	13

In this report the terms "dive" and "profile" are used interchangeably to indicate the data collected by the HRP. The following sequence of events comprises a dive or profile:

* The HRP is prepared for deployment:
  The instrument's computer is programmed with the dive control parameters, descent weights are loaded, mechanical systems checks are completed, and the ship is positioned at the desired deployment position.

* The dive control software is started and the HRP is deployed:
  The HRP acquires and stores data during the descent. When the first of several criteria for dive termination is met, data logging is terminated, the descent weights are jettisoned, and the HRP returns to the surface.

* The HRP is tracked acoustically during its descent and ascent:
  The HRP transducer pings every 20 seconds on the descent, once per second while the weights are being released, and every 5 seconds on the ascent.

* The ship is positioned for recovery and the HRP is recovered:
  The ship waits for the HRP to surface about a half mile downwind of the deployment position. Once the HRP is sighted, the ship steams slowly by and recovery lines are attached to the profiler using long poles with hooks at the ends. The HRP is then brought back on deck with the aid of the hydraulic lifting rig.

* The data are offloaded to a PC while the HRP is prepared for the next profile:
  The HRP can only store the data from one profile in its memory, so all the data must be offloaded between profiles using a "fast" (57.6 Kbaud nominal) serial data transfer program for PC (written by Jim Doutt at WHOI).

The duration of the above sequence of events varies with the depth of the profile. A 5000–meter dive takes about 4.5 hours to complete (descent 2.25 hours, ascent 1.25 hours, and download 1 hour). The ship usually moves to the next station during the data transfer, to minimize the time between deployments.

Once the data are on the PC, the files are transferred to a UNIX workstation via FTP for processing. The processing steps completed on the workstations are those that will be discussed in detail in this report. The order of downloading and processing has been optimized for speed.

A schematic of the fastest way to transfer and process the data is shown in figure 2.

The evaluation of the quality control (QC) plots of uncalibrated data from each sensor is very important. If the QC plots indicate sensor problems, they must be corrected before the HRP is re–deployed. The rest of the data processing is completed either during the download to PC or while the HRP is in the water obtaining the data for the next profile.

The HRP data are archived to exabyte (8 mm) tapes on the workstations daily. The raw data (and its byte–swapped version), the intermediate products, and the combined data are saved. The raw data are also archived to WORM optical disks on the PC, in case of serious malfunction of the workstations. We hope this won't happen, but at sea, redundancy is a sensible precaution.

HRP raw data format

Verification of the raw data quality is an important diagnostic tool for use with the HRP. Checking the raw data requires familiarity with tools for displaying binary data, PC and UNIX architectures, hexadecimal encoding, and the appropriate values for each sensor. The descriptions in the following section are intended to aid in the decoding and interpretation of raw HRP data files.

During a dive, the HRP writes data to its on–board memory as it samples. The unprocessed counts from the sensor channels are stored as binary (unformatted) data to optimize the usage of memory. The data in the HRP's memory must be transferred to a file before they can be examined. On a PC, the Norton Utilities "nu" program can be used, and on a Unix workstation, the command "**od –x finxxx.raw | more**" displays the file.

It is important to remember that the data was created on a PC, so data words are organized with the least significant byte first. If a fine raw file on the PC had 9b00 as its first pressure word, the steps for conversion to approximate scientific units would be:

swap bytes : 9b00 becomes 009b
convert to decimal : 009b hex == 155 dec
divide by 10 to obtain the approximate pressure of 15.5 db

Three kinds of data are collected during a profile: fine, micro and altimeter. A separate transfer to PC is made for each type of data stored in the HRP. For each type of data, a separate file is created, with a 1024–byte header, followed by all the data records for that type of data. The header contains 512 bytes of data describing the settings for the dive, followed by 16 bytes of scaling data for each of the 16 sensors (256 bytes total), followed by 256 bytes of filler. The include file (hdr2.h) used by the programs that read the raw data documents the exact header format and is listed in Appendix 1.

HRP at–sea data transfer and processing sequence

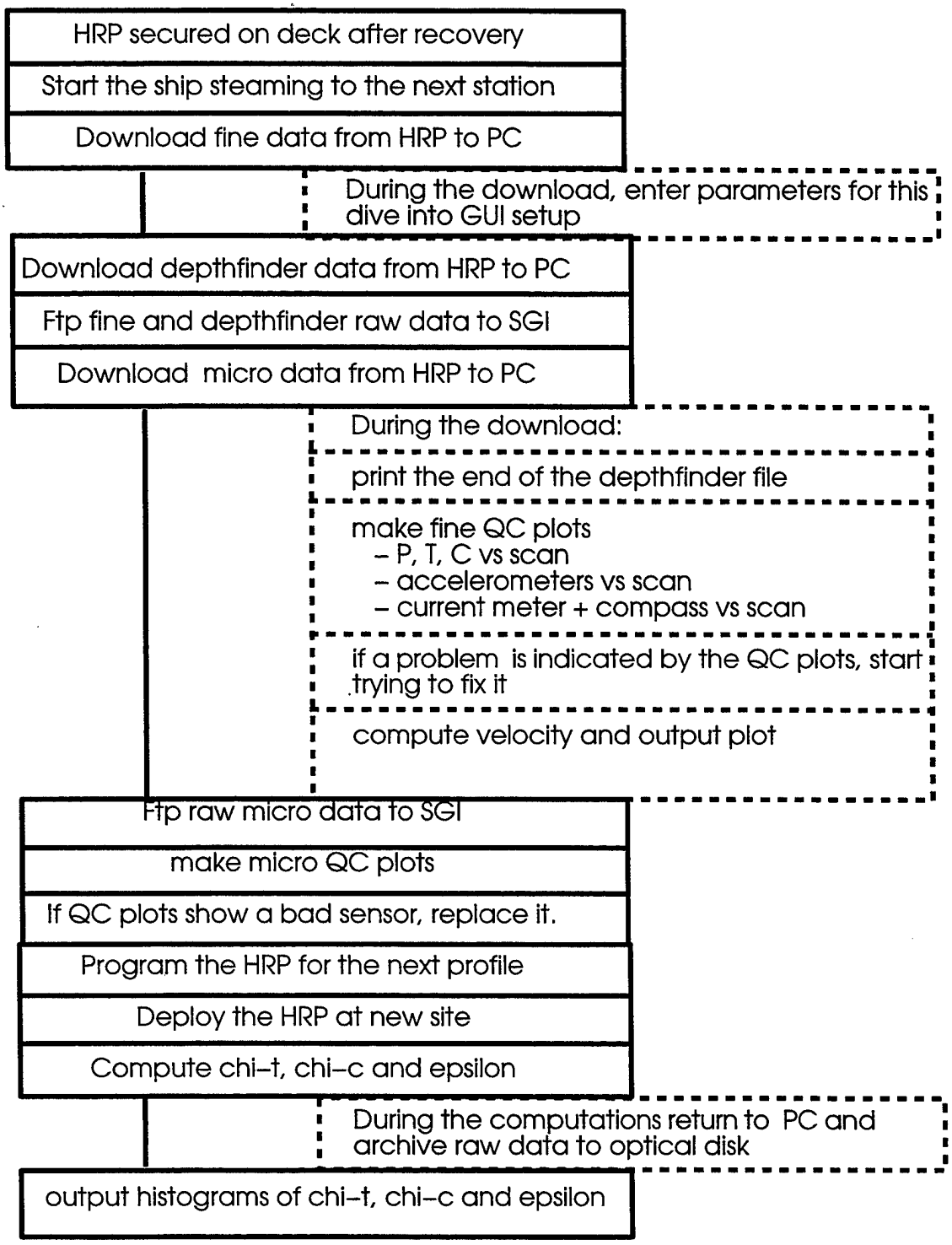


Figure 2 : Sequence of steps used in the optimized at–sea data transfer and processing system

A logical record for fine data is composed of 12 signed or unsigned integers (depending on the channel) acquired from sensors sampling at 10Hz. A sample of the **"od –x"** output of four records from a finescale raw file is shown below. The format of the output is an octal address followed by 8 short (2–byte) integer words, so 1.5 lines are required to display one finescale record. The first word in each record is bold below to help distinguish between records. These data are **NOT** byte swapped. Numerous lines of header (1024 bytes) were skipped so that only the beginning of the actual fine data is displayed here :

```
0010000 9b00 d145 9c82 7e00 93f3 8c0c ce03 7707
0010020 0ffe c7e3 2e0d 1000 9a00 d045 9c82 f605
0010040 01f3 f30a f105 1507 4efe cce3 330d 1000
0010060 9b00 cf45 9c82 1809 86f5 6b09 4006 9806
0010100 c3fe dde3 350d 1000 9b00 d545 9c82 2308
0010120 e0f9 4908 5904 f805 bdfe f3e3 450d 1000
```

For clarity, the fine data have been reformatted below to display one logical record per line. Labels for the sensor association with the columns have also been added:

	CTD			Accelerometers				Velocimeter		Compass		Ground
addr	PR	TE	CO	TAx	TAy	BAx	BAy	Vx	Vy	Cx	Cy	GR
0010000	9b00	d145	9c82	7e00	93f3	8c0c	ce03	7707	0ffe	c7e3	2e0d	1000
	9a00	d045	9c82	f605	01f3	f30a	f105	1507	4efe	cce3	330d	1000
0010060	9b00	cf45	9c82	1809	86f5	6b09	4006	9806	c3fe	dde3	350d	1000
	9b00	d545	9c82	2308	e0f9	4908	5904	f805	bdfe	f3e3	450d	1000

A micro record is comprised of four unsigned integers from sensors that are sampled at 200 Hz. Consequently, two logical records are displayed per line for the micro data. An example of the **"od –x"** output of eight records from a micro raw file is shown below. These data are **NOT** byte swapped. Numerous lines of header (1024 bytes) were skipped so that only the beginning of the micro data is displayed here :

addr	MC	MT	SX	SY	MC	MT	SX	SY
0010000	2330	b4ff	5846	dc2d	2630	adff	c003	dd2c
0010020	2b30	bbff	8eb9	7bb9	2a30	b2ff	452d	3103
0010040	2730	a9ff	2429	9f42	3030	beff	9bb6	eacc
0010060	3730	beff	0905	4bd9	3630	afff	3341	c246

The altimeter file contains the ranges from the bottom acquired at 1 Hz by the depthfinder. The ranges are stored as character strings, which is how they are received from the altimeter. The **"od –c"** output of a depthfinder file made during dock tests is shown below:

```
0010000   1   8   .   2   6   .   5   8   .   7   1   8   .   5   1   8
0010020   .   4   1   8   .   4   1   8   .   4   1   8   .   3   1   8
```

```
0010040   .  4  1  8  .  4  1  8  .  3  1  8  .  2  1  8
0010060   .  2  1  8  .  1  1  8  .  1  1  8  .  2  1  8
0010100   .  0  1  8  .  0  1  7  .  9  1  7  .  9  1  8
0010120   .  0  1  7  .  9  1  7  .  9  1  7  .  7  1  7
0010140   .  8  1  8  .  0  1  8  .  2  1  7  .  6  1  7
0010160   .  5  1  7  .  5  1  7  .  3  1  7  .  1
```

* The first column is the octal byte address in the file – the other columns are the character representations of the data.

** The data should be read left to right along the rows – the ranges shown here decrease from 18.2 to 17.1.

*** The second two ranges are usually bad (6.5 & 8.7) – they regularly lose a character in transmission – the rest of the data is good.

In preparing for a dive, it is important to consider the spatial requirements for data storage. The HRP has only 16 Mb of memory available to be used, and the segments assigned to fine and micro data storage must be allocated correctly for successful data logging. During one minute of data acquisition, the HRP stores 14.4 Kb of fine data, 96.0 Kb of micro data, and 0.36 Kb of altimeter data.

fine data : (60 seconds/minute x 10 records/second x 24 bytes/record) = 14.4 Kbytes
micro data : (60 seconds/minute x 200 records/second x 8 bytes/record) = 96.0 Kbytes
altimeter data : (60 seconds/minute x 1 record/second x 6 bytes/record) = 360 bytes

For a profile taking 140 minutes (about 5000 meters) about 2 Mb of fine data and 13.5 Mb of micro data is logged. The HRP's 16 Mb memory is addressed as blocks 0 to 255, with the header always written to block 255. Using 221 blocks to store micro data, and 32 to store fine data maximizes the possible memory usage. Altimeter data will always fit into the block reserved for it at address 254.

HRP Data Processing Software

In order to clarify the sequence and interaction of the programs used to process HRP data, diagrams of the old and new systems are presented on the following pages. Figure 3 shows the old VAX system and figure 4 shows the new SGI system. The following conventions are true for both figures:

–arrows indicate the data flow
–fine data processing flow has a grey background

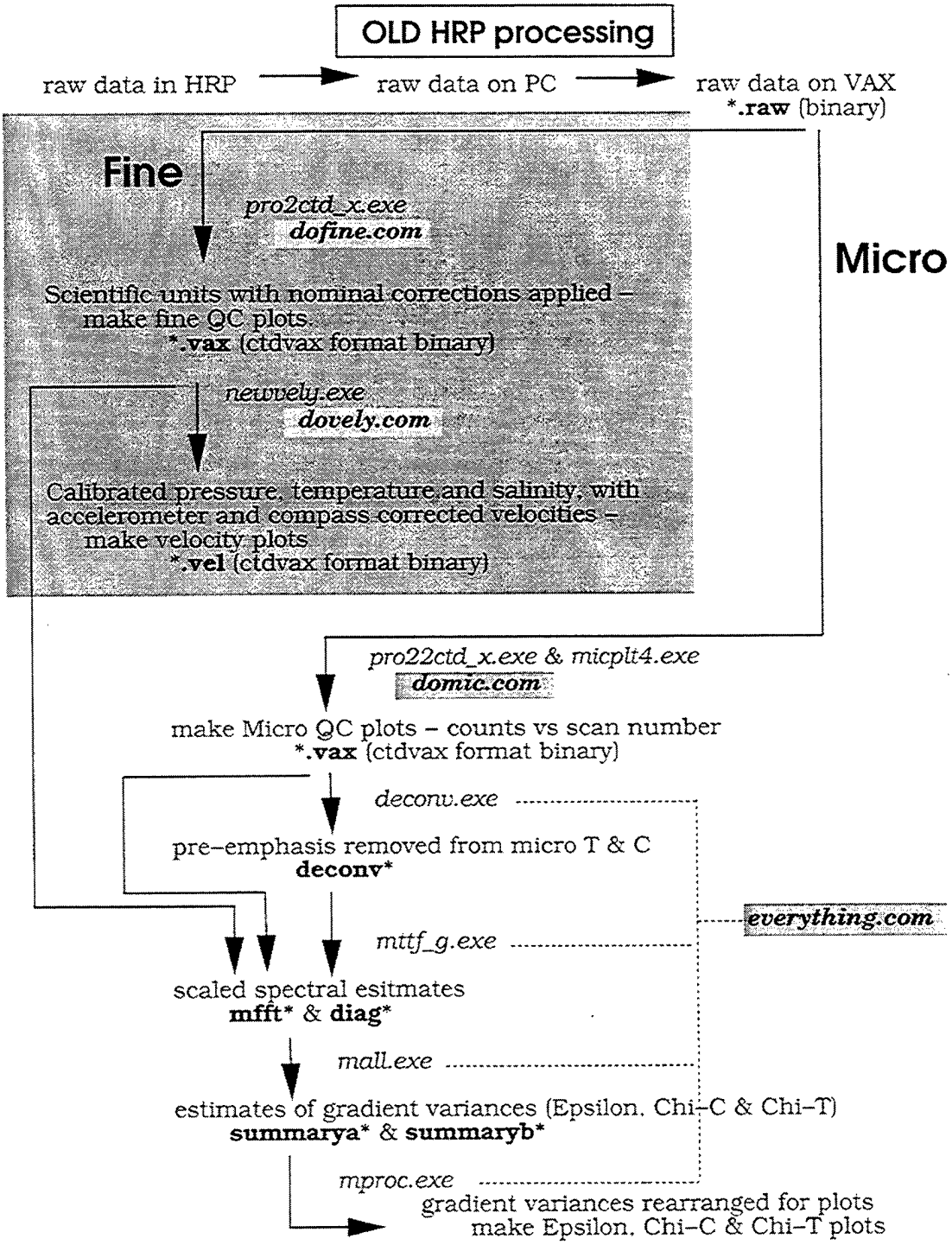


Figure 3 : Schematic representation of the original VAX based HRP data processing system that was used at sea.

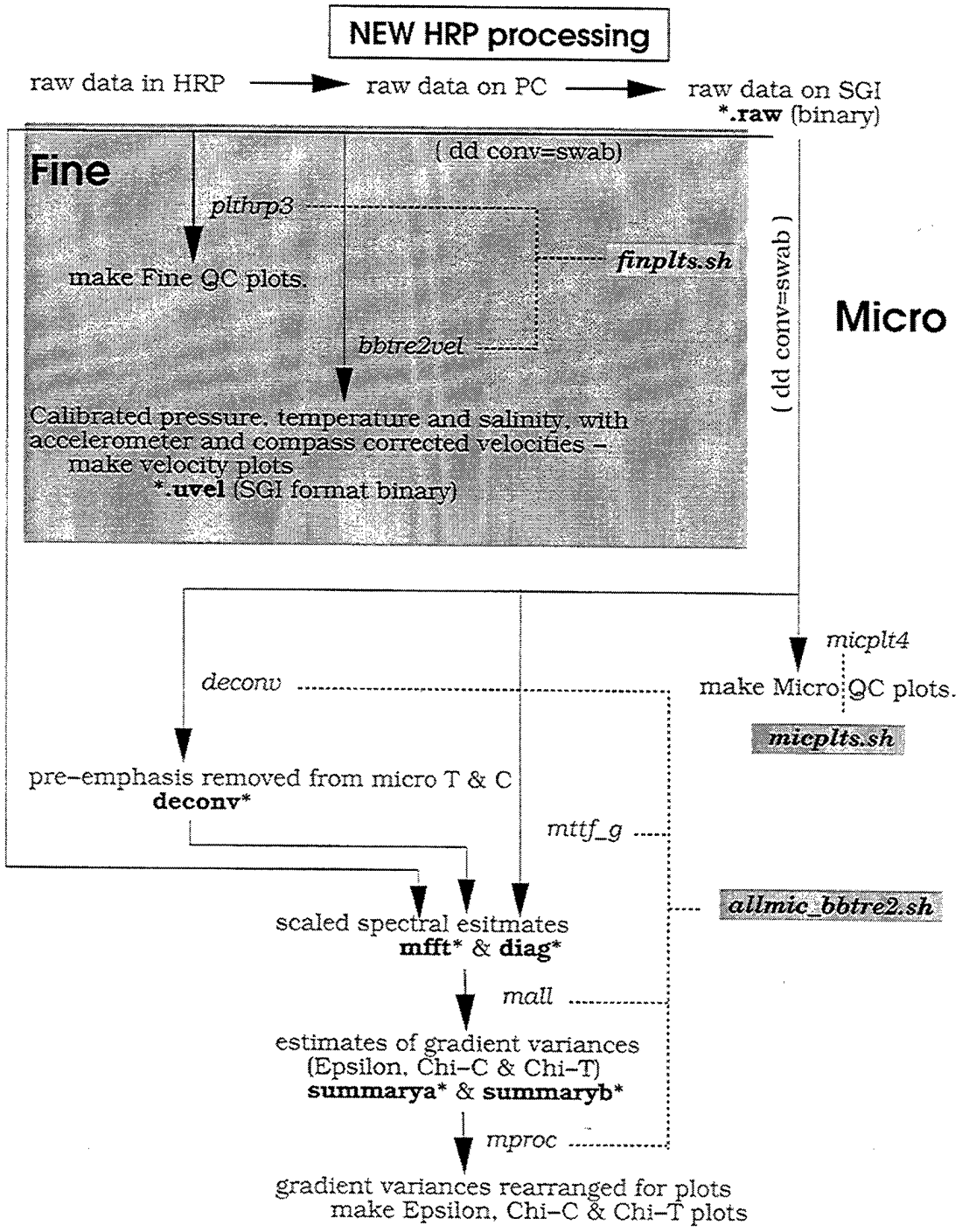


Figure 4 : Schematic representation of the new SGI based HRP data processing system that is currently used at sea.

–output file names are shown in bold type
–script or command file names are italicized bold on a grey background
–dotted lines connect programs run by a single script or command file

For example, the mfft_g program uses several kinds of inputs, indicated by the three arrows next to the mfft_g label. By tracing the arrows back to their origins, the data types required to run the program can be determined. The mfft_g program outputs spectral estimates in files with names starting with "mfft" and "diag", and is run by a shell script called allmic.sh or a command file named everything.com.

A few notes on the current HRP group directory structure and naming:

– Because the microstructure data files are so large, they are stored on a separate disk (/sea2). The fine data and the programs are stored on another disk (/sea1).

– The Fortran programs executed by the shell scripts reside in directories named after the program under /sea1/prgs/hrp.

– The development directory for the shell scripts is /sea1/prgs/hrp/hrp_proc. The final versions used on the most recent cruise are in /sea4/hrp/scripts.

– All data for a cruise are stored in directories by type under a directory named for the cruise based on the ship name and cruise ID. HRP is the main type, but there may also be ctd, adcp, bathy or other kinds of data stored under that cruise's directory.

– HRP raw and velocity file names include a letter indicating which cruise they came from incorporated into the names. The name–cruise association is done alphabetically with chronology, so, micb006.raw contains raw microstructure from the second HRP cruise (OC218), and finh047.uvel, contains velocity data from the eighth HRP cruise (SJ9701).

Generally the HRP data will be processed using the GUI tool, so most users will be unaware of the details of which programs run and how they interact. However, if changes to the processing system are required, the detail provided here will be useful background.

FORTRAN Programs

The following is a brief description of each of the FORTRAN–77 programs routinely used in the current HRP data processing system used at sea. These summaries provide background for

understanding the data flow, as shown in figure 4. Remember that "dd" must be used once to swap the bytes in a raw data file and create the "_s" file before additional processing occurs, and that pro2ctd_x is not used in the new processing system. In the naming scheme used below, * refers to the part of the name of the file that includes the profile number. For instance, fin*_s.raw could indicate finh045_s.raw and mfft* could refer to mfft_g.d038.

plthrp3 inputs : any fine scale data:
(fin*_s.raw, fin*.vax, fin*.vel (from old system) or fin*.uvel files)
outputs : ascii listing or postscript plot file

This is a versatile program for displaying or listing HRP data, and was developed from proplotx on the VAX. The calls to CalComp plotter routines were replaced with calls to GMT's postscript library for output. The current version reads _s.raw files, ctdvax files (for compatibility with old data) or .uvel files. It can also compute parameters like density and theta from the existing variables. The variables chosen can be either displayed as an ascii list or output as postscript format files. This program is the one used to make the fine QC plots and the velocity plot.

nhrpvel11, or ***bbtre2vel*** input : fines cale fin*_s.raw file
output : fin*.uvel file

These programs are all based on newvely, with slight modifications to accommodate software or hardware enhancements. Nhrpvel11 was the version used for BBTRE1. The Acoustic Current Meter head was modified prior to BBTRE2, and was wired orthogonal to what the previous one was, necessitating the changes incorporated in bbtre2vel.

In either case, the accelerometer and compass data are used by this program to convert the relative velocities measured by the HRP's orthogonal transducers to absolute velocities (east and north components). The velocity data are output with pressure temperature and salinity toa plain binary file (.uvel). These data are later combined with the diffusivities to generate the final data product which is called a combined file.

micplt4 input : microscale mic*_s.raw
output : plot1.dat & plot2.dat

This program is used to read the microstructure raw data, apply some calibrations, and arrange it in order to make a QC plot. A script for the PlotPlus graphing program called micqc_tmpl reads the plot*.dat files created by micplt4 and generates the micro QC–plot.

deconv input : microscale mic*_s.raw file
output : deconv_* files

The pre–emphasis of the Micro–T and Micro–C channels is removed by this program. It is the first step in the microstructure processing. Documentation of the principals behind the software used for processing microstructure data from the HRP is provided in the paper by Polzin and Montgomery(1996).

mfft_g inputs : fine and micro *_s.raw files, deconvolved T & C files
outputs : mfft* and diag* files

This program scales and transforms the data, then obtains the spectral estimates for all the microstructure channels. This program is the main component of the microstructure processing.

mall inputs : mfft* and diag* files, proc.dat
outputs : summarya* and summaryb* files

The spectra made by mfft_g are integrated to the point where the gradient variance estimates are made. The program requires that a parameter file called proc.dat exist to provide the integration limits. The summarya.d### file contains the segment number, pressure intervals over which the computations were made, drop rate for that interval, and the data quality word associated with each channel's data (1 is good, 0 is bad). The summaryb.d### file contains the segment number, Chi–T, Chi–C and two independent estimates of Epsilon. The data in the summary files are combined with the output of the velocity computation program (bbtre2vel) to create the combined file. This is the final data product.

mproc inputs : summarya* and summaryb* files
outputs : mic*, pdf* and diag* files

This program reformats the good gradient variance estimates so that they can be plotted as histograms of Chi–T, Chi–C and epsilon vs pressure. These plots are helpful in quantifying the levels and depths of mixing observed in an HRP profile.

Shell Programs

On the UNIX workstations, Bourne shell programs are used to automate the execution of Fortran programs listed above that comprise the conversion and analysis software. These scripts require that the following cruise–specific environment variables be set by each user:

```
setenv  CRUISE          sj9701
setenv  FINE_PATH       /sea1/data
setenv  FINN            sea1
setenv  MICRO_PATH      /sea2/data
setenv  MICR            sea2
setenv  WORK_PATH       /sea4/hrp/work
setenv  PROC_PATH       /sea4/hrp/scripts
setenv  CRUIS_LETTER    h
```

Having the directory paths in environment variables allows for easier transitions between cruises, because it saves having to hand–edit each program to modify the paths. The $PROC_PATH variable points to the directory where the shell programs for the cruise reside, and $WORK_PATH points to the directory to which temporary files are written.

It is assumed that directory structures of $FINE_PATH/$CRUISE and $MICRO_PATH/$CRUISE (/sea1/data/sj9701 and /sea2/data/sj9701) exist for the shell scripts' use in reading and writing data.

Template files are employed to provide the correct input to each program's queries. The templates have large numbers like 8888 in all station–specific fields. The UNIX "sed" utility replaces the 8888's with the dive–appropriate numbers just before the program is executed. The "sed" output with the correct dive–specific parameters is written to a temporary file in $WORK_PATH (/sea4/hrp/work) and used as input when the program is executed. An example template file that gets modified to provide input to plthrp3 is shown in Appendix 2a.

The shell program that controls the creation of the fine quality control (QC) plots and runs the velocity computations is called finplts.sh, and is listed in Appendix 2b. This script performs the following tasks:

- swaps the bytes of the raw fine input file
- runs plthrp3 to generate the following QC plots:
    + uncalibrated pr, te, and co vs. scan number
    + accelerometer counts vs scan number
    + raw velocimeter and compass counts vs scan number
- prints the plots
- executes n_hrpvel to compute corrected velocities
- runs plthrp3 to generate the plot of :
    + temperature, salinity, east and north velocities vs pressure
- prints the plot

In order for finplts.sh to work correctly, the following template files are required to be in

$PROC_PATH (/sea4/hrp/scripts):

dopl_ptc	njmtvel_tom.com
dopl_accs	dopl_velts_n
dopl_vlcm	

The script that makes the microstructure QC plots is micplts.sh; it does the following:

- swaps the bytes of the raw micro input file
- runs micplt4 to reformat the data for plotting
- uses PlotPlus software to make the micro QC plot
- prints the plot

The template file dopl_mic is required to be in $PROC_PATH (/sea4/hrp/scripts) for micplts.sh to run.

There is also a script called doplts.sh that creates both the fine and the micro QC plots. This script is useful for reprocessing profiles, but not during the initial processing, when the micro raw data is not transferred until long after the fine raw data are available.

Allmic_bbtre2.sh is the script that runs the microstructure processing sequence that computes the diffusivities and turbulence estimates. Allmic_bbtre2.sh performs the following tasks:

- executes deconv on Micro–T and Micro–C
- executes mfft_bbtre2
- uses PlotPlus to make a diagnostic plot
- executes mall_bbtre2
- executes mproc_sgi
- uses PlotPlus to make the histograms of Epsilon, Chi–T and Chi–C vs pressure

A very large and cumbersome program would be required to encompass all the functions described above. Consequently, allmic_bbtre2 was organized to make calls to other scripts that perform processing steps associated with single programs. Allmic_bbtre2.sh is listed in appendix 3a. The shell scripts called by allmic_bbtre2.sh are: deconv.sh, mfft_bbtre2.sh, diagplt.sh, mall_bbtre.sh, and mproc_sgi.sh; and are listed in appendices 3b– 3f, respectively. Appendix 3g contains the code for alchieps.sh, the script that is called by mproc_sgi.sh to generate the histograms.

In order for allmic_bbtre2.sh and its associated scripts to work correctly, the following shell scripts and template files are required to be in $PROC_PATH (/sea4/hrp/scripts):

deconv.sh	mproc_sgi.sh
decon_c.inp	do_mp_sgi
decon_t.inp	alchieps.sh
diagplt.sh	cc_sgi_tmpl

mfft_bbtre2.sh	ct_sgi_tmpl
mfft.inp	ep_sgi.tmpl
mall_bbtre2.sh	PROC.DAT

The preceding description of the shell programs and their functions may seem complicated, but having the scripts control the processing has significantly lessened the operator intervention required for running the programs. Only one or two people need to know all the details, for the rest of the users, the system is much simpler.

Description of the Graphical User Interface (GUI)

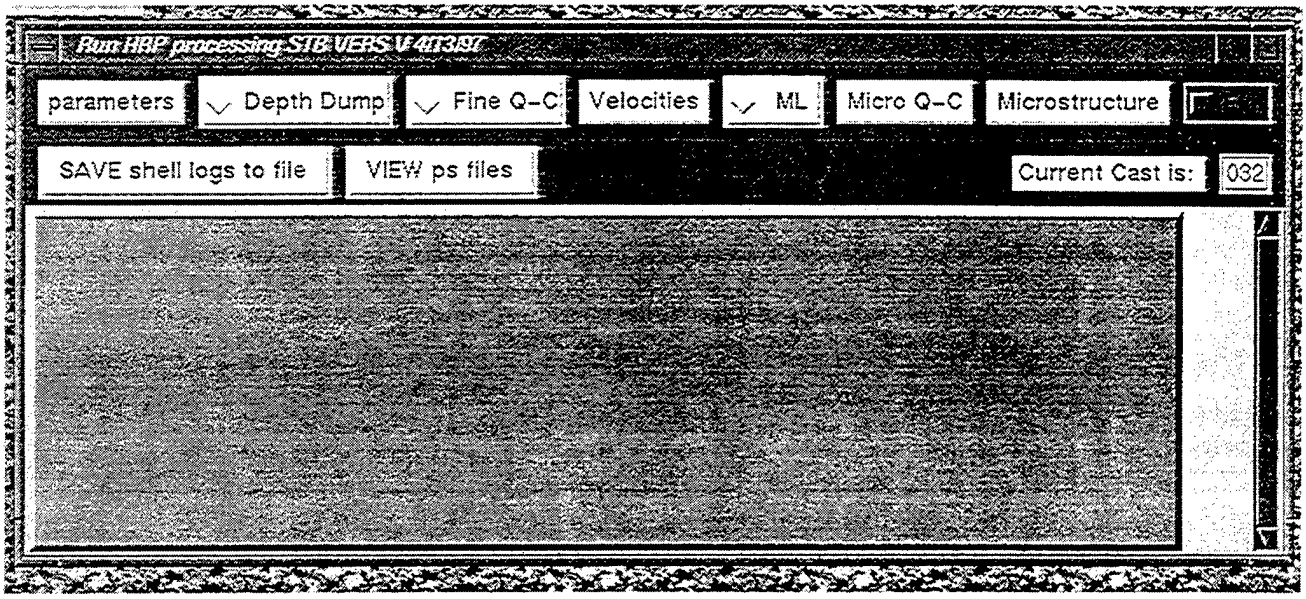
The GUI tool was designed and implemented on the Brazil Basin Tracer Release Experiment year 1 cruise, and enhanced during the year 2 cruise. The intention was to simplify the method of processing HRP data at sea. Previously, the user had to know a lot about how the the processing worked to run it, now the user just has to click on buttons and enter some parameters. The GUI interface made this software much easier to use. The GUI tool handles errors reasonably well, so users should experiment with it, if the necessary action or input is unclear.

For the GUI tool to work, the following environment variables must be set, in addition to the cruise specific ones listed in the previous section.

```
# for GUI applications to work
setenv TCL_LIBRARY    /usr/local/tcl/lib
setenv TK_LIBRARY     /usr/local/tcl/lib
```

The GUI program for processing the HRP data is called **dohrp5** and was developed using the Tk/Tcl language. Tcl is a scripting language written by Osterhout (1993) and first released in 1991. Tk is the complimentary library of GUI elements that allow easy creation of windows, menus and buttons. Together they provide the raw materials for building an interface in which clicking on a button invokes a shell script that performs some function. Tk/Tcl was chosen for this project because it is shareware, has a large user base, and works on a variety of platforms (UNIX, Windows, Mac). The implementation described here uses Tcl version 7.4 and tk version 4.0.

When **dohrp5** is invoked, a two part window is displayed, as shown in figure 5a. Processing options are listed as menu items along the top of the window and space for status messages occupies the bottom. As functions are chosen from the main menu, new windows for those functions are automatically opened below the main window or overlapping it.



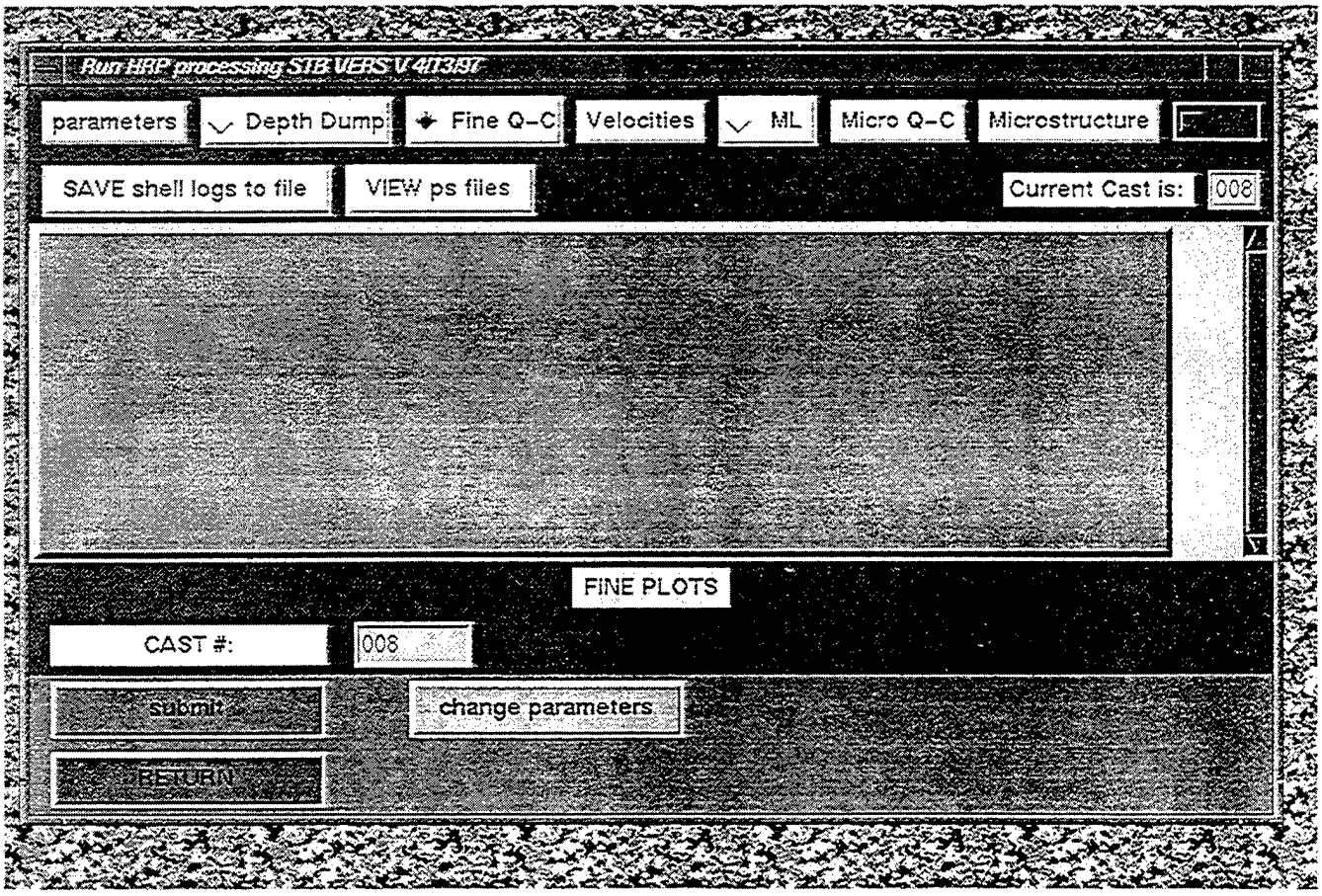


Figure 5 **a)** The main menu of the **dohrp5** GUI tool at startup
b) The **dohrp5** window, with sub–window for making fine QC plots

The key to accomplishing the HRP processing with the GUI tool is to work from left to right across the top line of menu options, and then move to the second line if needed. The menu items each correspond to a different component of the HRP processing, organized in a temporal sequence appropriate to at–sea data processing. The items with diamonds preceding them perform a single function. The ones without diamonds have more than one function choice associated with them. Menu options should always be invoked independently from the main menu. For instance, if a Fine QC plot was made, the fine QC sub–window should be closed using the "return" button before another function is chosen from the main menu.

The first step in the sequence is setting up the parameters associated with a profile. Clicking on "parameters" (the left–most menu item) presents the following options:

set cast number
show or change parameters
get parameters
set the input parameter file
save this cast's parameters

At the beginning of a cruise, or to work with data from a different experiment, the "set input parameter file" option must be chosen to specify the file to be used to obtain or store the parameter data accessed by the processing programs. The default parameter file is /sea4/hrp/scripts/n_hrp.dat. This option only needs to be used once: the file selected is used for the whole session, unless explicitly changed.

The first few lines of the parameter file used during the BBTRE2 experiment are shown below:

```
sta  pbias     cbias     smin  smax   sx     sy    ml pmax usemin usemax decl
001 -19.170 -0.01410 05000 10000 29.28 33.99 90 1000 1 100000 24
002 -22.964 -0.01410 10000 80000 29.28 33.99 75 5159 1 100000 24
003 -21.866 -0.01410 10000 80000 29.28 32.17 30 5415 1 100000 24
004 -22.065 -0.01410 10000 80000 29.28 32.17 40 5323 1 100000 24
005 -19.669 -0.01410 10000 80000 22.99 32.17 55 5383 1 100000 24
006 -22.365 -0.01410 10000 80000 22.99 32.17 60 5431 1 100000 24
007 -22.065 -0.01410 10000 80000 32.17 22.99 45 5407 1 100000 24
008 -22.165 -0.01410 10000 80000 32.17 25.66 45 5105 1 100000 24
009 -22.365 -0.01410 10000 80000 32.17 21.91 50 5267 1 100000 24
```

The columns above are: profile number (sta), pressure bias (pbias), conductivity bias (cbias), the scan range over which to compute the mean velocity (smin, smax), calibration factors specific to the shear probes used on that dive (sx, sy), the pressure at the bottom of the mixed layer (ml), the maximum pressure (pmax), the range of scan numbers to use (usemin, usemax), and the declination of the location of the dive (decl). This file is not format sensitive; any white space

character separating the fields is accepted, but there can only be **one** tab or space between each field.

Having a parameter file that contains all the input variables has been a great help in documenting what values and options were used to process a profile. The data in n_hrp.dat should be saved to another name at the end of a cruise so the processing parameters actually used at sea are not overwritten accidentally.

The next step is preparing to process a new dive. Executing the following procedure, using the options under the parameters menu, is required:

1– click on **set cast number** and enter the new number, then click on "accept this cast"
2– click on **show or change parameters** (–9999.99 signifies that data have not yet been entered)
3– enter the parameters needed (not all the parameters are required for every function): scan range is not needed until the velocity computation, and shear probe calibration values aren't needed until running mfft.
4– click on **save** to write the parameters to the parameter file selected

The procedure for viewing the existing parameters for a profile is similar to the above, except that before the second step, "get parameters" would be used to read in the data from the parameter file, and steps 3 and 4 would not need to be done.

Once the parameters pertaining to the dive to be processed are entered, the rest of the sequence is simple. Again, work through the options from left to right across the top line.

All of the menu items that allow execution of processing programs (depth dump, fine Q–C, Velocities, ML, Micro Q–C and Microstructure) have the same window format when invoked. Figure 5b shows the basic format of the new sub–window created as a result of choosing "fine Q–C". The top part of the new window shows the input parameters to be used for that function on a dark background. Other values may have been entered, but only those pertaining to the function are displayed. Since making the fine Q–C plots only requires the cast number, it is the only value displayed. Below the input parameters are three buttons on a lighter background: **submit** is green, **return** is red, and **change parameters** is beige. Clicking on "submit" causes the script associated with the function choice to be executed, clicking on "return" closes the sub–window, returning focus to the main window, and clicking on "change parameters" displays the window with all the parameters possible as shown in figure 6. If a change is made to the parameters, it will be used to execute the program selected, but the "save" option under the parameters heading must be used before the change is made permanent.

Finally, to start processing a profile, select the "depth dump" button to invoke a script that prints a list of the unformatted ascii strings saved from the depthfinder. For profiles terminated by

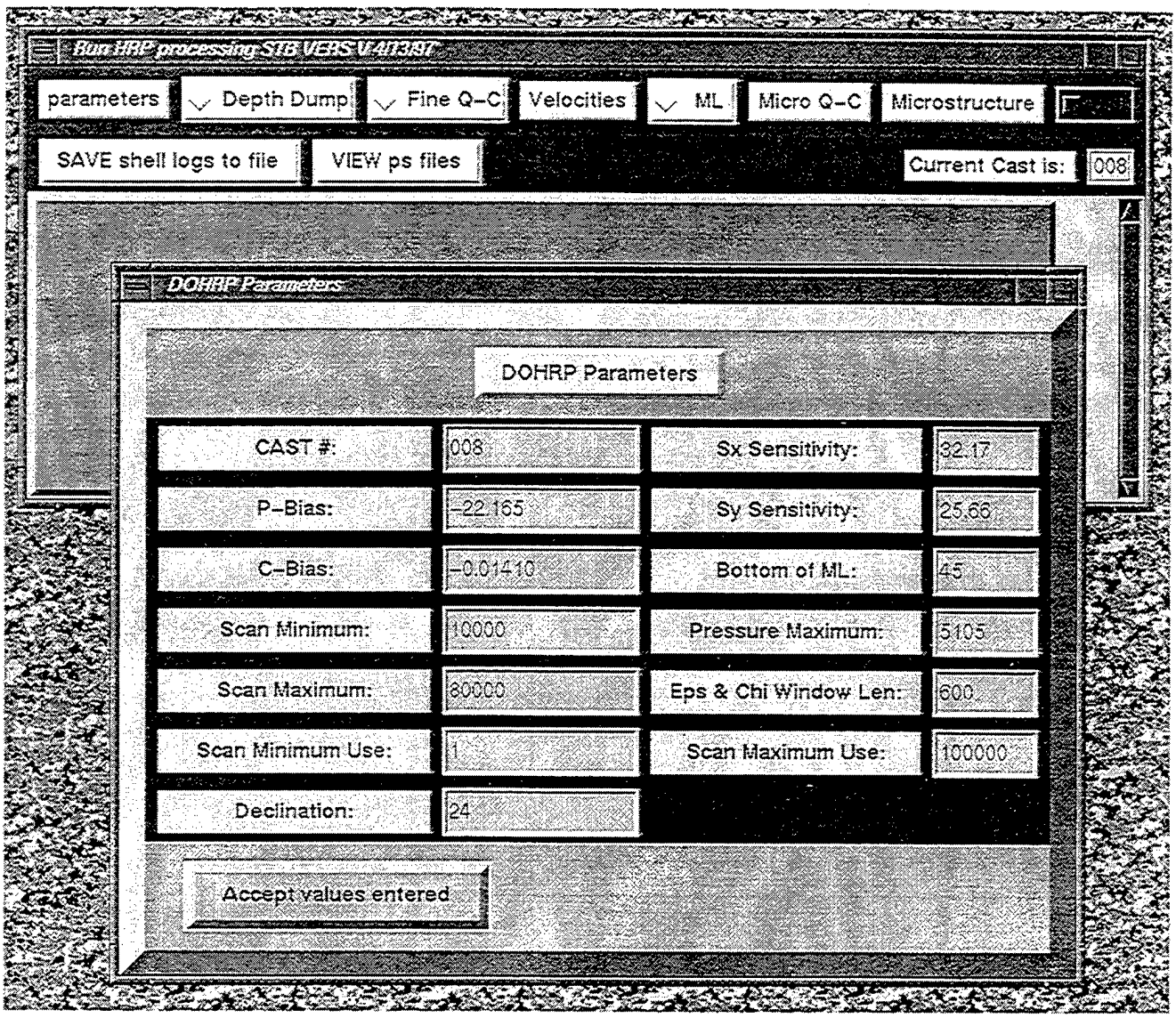


Figure 6. The main **dohrp5** window with overlapping parameter display window.

pressure, this listing is used in determining how far the HRP was from the bottom at dive–end. Normally, outputting the altimeter data precedes making the fine Q–C plots, but it doesn't have to. The GUI tool is flexible; so that the altimeter data from several profiles could be printed sequentially by changing the cast number and clicking on the "submit" button after each change.

Generating the fine quality control plots comes next in the sequence. Choosing the "fine Q–C" option causes a small second window to be appended to the existing main window (see figure 5b). This option executes a script that byte–swaps the raw data file and prints the uncalibrated scan vs channel plots described earlier in this report.

The "velocities" menu option invokes the program that applies all the calibrations to the raw data, and computes salinity, drop rate, and east and north velocity components. The program's output, the .uvel file, is the source of the corrected finestructure variables.

Under the "velocities" menu option, there is a choice that allows nodding corrections to be applied or not. Nodding corrections adjust the velocity data additionally for the profiler's movement as it descends. Normally nodding corrections are applied, but to aid in analysis, the data from BBTRE1 were processed without making the nodding correction. Both options execute the same program, currently bbtre2vel: the difference is which input template file is employed.

Next, the depth of the temperature mixed layer needs to be determined. Choosing the "ML" option starts Matlab and executes a program that displays the temperature profile for the top 100 meters of the ocean. The plot is displayed for 15 seconds, so the depth of the bottom of the mixed layer must be chosen during this interval. The mixed layer depth is required by the microstructure computations, so it must be added to the parameter list (click on show or change under the parameters menu, change the value, and then save) before starting the microstructure computations.

When the microstructure data transfer from the PC is completed, the micro quality control plots can be made. The "micro Q–C" option functions similarly to the one for making the fine QC plots, but gives you the option of extending the scan number axis by making four pages of plots instead of fitting the whole scan number range on one sheet. This allows a more accurate assessment of the data quality.

The steps of the microstructure processing are accessed from the microstructure menu and are listed here :

ALL, SGI
deconv, SGI
mfft, SGI
mall, SGI

mproc
print plots

Usually the "ALL SGI" option is used to run the script that performs the whole processing sequence, including outputting plots. If however, only part of the process is to be done, the options could be used separately. This allows flexibility in reprocessing only the stages that need to be done, instead of recomputing absolutely everything. The options under microstructure are organized in vertical order, with the first step at the top. Refer to figure 2 to check that the necessary input files exist before choosing to execute one of these programs independently.

The options on the second row of the main menu are less used than those on the first line. The **"save shell logs to file"** option allows the text written to standard out window to be saved to a file. This is often helpful in determining the source of a computation problem. The **"view ps files"** option lets you display the postscript files from the GUI tool, which is done solely for ease of use.

Finally, pressing the red "exit" button at the top right corner of the main window terminates the **dohrp5** program, and closes any associated windows that were open.

Summary

The conversion of the programs used to process HRP data for use on UNIX computers provided an opportunity evaluate and enhance the functionality of the entire software system. The whole processing sequence was redesigned and optimized, often by employing the added functionality of the UNIX utilities. The more streamlined processing system, combined with using faster computers has resulted in significantly decreased processing time. The GUI interface has made processing data from the HRP much easier to accomplish. The routine analysis of HRP data at–sea is now completed faster and easier than before, so the software conversion was definately a success.

References–

Montgomery, E. T. (1991) High Resolution Profiler (HRP) user's guide and software modifications documentation. *Woods Hole Oceanographic Institution Technical Report* WHOI 91–01, 32 pp.

Osterhout, John K. (1993) TCL and the TK Toolkit, Addison Wesley Publishers.

Polzin, K., and E. T. Montgomery (1996) Microstructure profiling with the High Resolution Profiler (HRP). *Proceedings of the Microstructure Sensor Workshop*, pp. 109–115.

Schmitt, R. W., J. M. Toole, R. L. Koehler, E. C. Mellinger, and K.W. Doherty (1988). The development of a fine- and microstructure profiler. *Journal of Atmospheric and Oceanic Technology*, 5(4), 484–500.

Schmitt, R. W., E. T. Montgomery, and J. M. Toole (1995). A free–vehicle explores deep–sea mixing. *Oceanus*, 38 (1), 21–25.

Wessel, P., and W. H. F. Smith (1991). Free software helps map and display data, *EOS Transactions, American Geophysical Union,* Vol. 72, pp. 441, 445–446.

Wessel, P., and W. H. F. Smith (1995) New version of the Generic Mapping Tools released, *EOS Transactions, Ameican. Geophyical Union*, vol. 76, pp. 329.

Acknowledgements–

This work was made possible by funding from the National Science Foundation, grant number OCE–9415589. We thank Ray Schmitt and John Toole for their encouragement and patience with the implementation of this project. We appreciate Gwyneth Packard's and Barbara Gaffron's editorial assistance with this document.

Appendix 1 : Header record detail

The HRP data files are each made up of a 1024 byte header, followed by all the Fine, Micro or altimeter data records obtained during a profile. The header contains 512 bytes of data pertaining to the settings for the dive, followed by 16 bytes scaling data for each of the 16 sensors, followed by 256 bytes of filler. Fine, Micro and altimeter files for the same HRP profile have the same header. The include file (hdr2.h) used by the SGI programs that read the raw data is listed below. It shows all the variables in the header format and their size and order.

```
/*
* name: hdr2.h
* date: 16 Aug, 1994
*
* this include file provides the structure for the MICROFINE profiler
* header record. Made from prada's header.h for use in reading the PC
* data on the SGI. For use on this architecture, PC's int's had to
* become "short"s (etm)
*/

 /* totals 16 bytes : to be nested as part of header_c */
typedef struct sensor
{
        char mnem[2];           /* sensor mnemonic */
        short offset;           /* sensor offset */
        short mslope;           /* sensor slope multiplier */
        short dslope;           /* sensor slope divisor */
        short adchan;           /* a/d channel (if applicable) */
        short sfill[3];         /* sensor expansion */
}sens_type;

/* now the biggie.... */
typedef struct hd1
{

/* 256 bytes of ASCII info */
        char cast[4];           /* cast I.D.*/
        char ship[2];           /* ship name */
        char crus[6];           /* cruise number */
        char expn[52];          /* expansion */
        char com1[64];          /* comment line 1 */
        char com2[64];          /* comment line 2 */
        char com3[64];          /* comment line 3 */

/* GENBYTES  bytes of General Parameters : 256 bytes here to sensors */
        short day;              /* day of month */
        short month;            /* month of year */
        short year;             /* 4 digit year */
        short deglat;           /* degrees latitude */
        short minlat;           /* minutes latitude X 100 */
        short deglon;           /* degrees longitude */
        short minlon;           /* minutes longitude X 100 */
        short fsampl;           /* FINE sample rate in Hz */
        short msampl;           /* MICRO sample rate in Hz */
        short navper;           /* NAVG sample period in Hz */
```

```
        short umspres;              /* user-spec'd MICRO start pressure */
        unsigned short umepres;     /* user-spec'd MICRO  end pressure */
        short micblks;              /* no. of micro blocks */
        unsigned short ufepres;     /* user-spec'd FINE end pressure */
        short fshour;               /* FINE start hour */
        short fsmin;                /* FINE start minute */
        short fssec;                /* FINE start second */
        short fehour;               /* FINE end hour */
        short femin;                /* FINE end minute */
        short fesec;                /* FINE end second */
        unsigned short fspres;      /* FINE start pressure */
        unsigned short fepres;      /* FINE end pressure */
        unsigned short fstemp;      /* FINE start temperature */
        unsigned short fetemp;      /* FINE end temperature */
        short mshour;               /* MICRO start hour */
        short msmin;                /* MICRO start minute */
        short mssec;                /* MICRO start second */
        short mehour;               /* MICRO  end hour */
        short memin;                /* MICRO end minute */
        short mesec;                /* MICRO end second */
        unsigned short mspres;      /* MICRO start pressure */
        unsigned short mepres;      /* MICRO end pressure */
        unsigned short mstemp;      /* MICRO start temperature */
        unsigned short metemp;      /* MICRO end temperature */
        unsigned short dspres;      /* pressure to turn on depthfinder */
        short drange;               /* range to end cast */
        char exflag;                /* saves method of cast termination */
        char choice;                /* flag for shadowgraph vs depthfndr */
}header_a;

/*  when reading the structure as one big structure, on the SGI indigo, the data skipped fscanms.
This is because of how the padding of chars and shorts is handled.  Breaking the header into 3
parts where the unsigned ints occur is a bit of a kludge, but works OK
*/

typedef struct hd2
{
        unsigned int fscanms;       /* fine scan for microstructure start */
        short nsens;                /* # sensors per record */
} header_b;

typedef struct hd3
{
        unsigned int bytes;         /* bytes in file */
        int dbytes;                 /* bytes in depth finder file */
        char type[4];               /* file type FINE/MCRO/NAVG */
        short ifill[82];            /* expansion space */
        sens_type sens[16];         /* sensor fields --- 256 bytes total */
        short morefill[127];        /* fills the last 256 bytes of the 1024 byte header */
}header_c;
```

Appendix 2: Fine scale processing
a.) sample template file for use with plthrp3

This sample template file is used as input for plthrp3 to make the P,T,C vs scan number quality control plot. There are several other template files each of which generates a different type of plot.

The reason to have templates is versatility. The environment variables $CRUISE, $FINN, and $CRUIS_LETTER, replace "fff", "ccc", and "lll" respectively. The 888 will be replaced by the profile number which is provided as a command line argument. The script in the next appendix has examples of how "sed" is used to make the substitutions. The invocation used to run plthrp3 after making the needed substitutions would be: plthrp3 < dopl_ptc.inp.

To understand more thoroughly what the lines in the templates actually mean, run the program for which they provide input and respond to the questions based on the information in the template file.

```
#dopl_ptc
R                                       # read a raw file
N
Y
12
1,90000
888,0,0
/fff/data/ccc/hrp/raw/finlll888_s.raw   # input file name
0
99,0,1,'SCAN'                           # x axis setup
0,90000,12500,10000,2
N
Y
4                                       # number of y axes
12,0,1,'GR'                             # y axis setup
-60,120,30,30,4
3,0,1,'SA'
28,58,5,5,2
2,0,1,'TE'
0,30,5,5,2
1,0,1,'PR'
0,5400,900,900,2
scan number
a/d Ground
Conductivity
Temperature
Pressure
0,0,0
```

Appendix 2: Fine scale processing
b.) shell script to modify the template and run plthrp3

Below is an example of a shell program that uses sed to modify a template file, the output of which becomes input to another program. The string 888 is replaced with the first argument. To make the fine quality control plots for profile 17, this script would be invoked as "sh finplts.sh 017". This is the script that is executed when the make fine plots button is clicked in the dohrp5 GUI tool.

```
#!/bin/sh
#
# finplts.sh
#
#  shell program to:
#   – byte swap the raw data and output the *_s.raw file,
#   – make 3 HRP fine quality plots :
#            (p,t,c on one, accelerometers on another, raw vels & compass on the third)
#
# The script can be run from any directory – it does not have to be run
# from where the programs exist, in fact it shouldn't be run from there!!!
#
#  emontgomery 10/94  (mod 9/95)
#
#   Tom Bolmer 3/13/97
#   moved all the scripts to /sea4/hrp/scripts
#    modified them to use $CRUISE and $FINE_PATH environment variables
#
#########################################################################

# check for presence of arguments, and exit if not given
if [ $# –eq 0 ]
then
 echo 'you need to use arguements: 001=stn #'
 echo '   for good filenames, please enter 3 characters for the number'
 exit
fi

#  set the umask so the protections will allow group access
umask 111

#=======================================================================
# do  the swapping of byte order now!
#   has to be executed from $FINE_PATH/$CRUISE/hrp/raw
cd $FINE_PATH/$CRUISE/hrp/raw
'echo  dd if=fin"$CRUIS_LETTER"888.raw of=fin"$CRUIS_LETTER"888_s.raw conv=swab \
   | sed s/888/$1/g'
cd $WORK_PATH

#=======================================================================
# now start making the plots
# the following is the equivalent of running dofine on the vax
#
echo 'generating finestructure data quality–control plots'
```

```
#  make grnd, c, t, p vs scan # plot
sed "s/888/$1/g; s/ccc/$CRUISE/g; s/fff/$FINN/g; s/lll/$CRUIS_LETTER/g" \
  $PROC_PATH/dopl_ptc > dotmp1
/sea1/prgs/hrp/pspropl/plthrp3 < dotmp1
 chmod 777 plot.ps0
mv plot.ps0 `echo 'finxxxptc.ps'|sed "s/xxx/$1/"`
lp `echo 'finxxxptc.ps' | sed "s/xxx/$1/"`

# make 4 accellerometer vs scan # plot
sed "s/888/$1/g; s/ccc/$CRUISE/g; s/fff/$FINN/g; \
    s/lll/$CRUIS_LETTER/g" $PROC_PATH/dopl_accs > dotmp2
/sea1/prgs/hrp/pspropl/plthrp3 < dotmp2
 chmod 777 plot.ps0
mv plot.ps0 `echo 'finxxxacc.ps'|sed "s/xxx/$1/"`
lp `echo  'finxxxacc.ps' | sed "s/xxx/$1/"`

# make velocimeter and compass vs scan # plot
sed "s/888/$1/g; s/ccc/$CRUISE/g; s/fff/$FINN/g; \
    s/lll/$CRUIS_LETTER/g" $PROC_PATH/dopl_vlcm > dotmp3
/sea1/prgs/hrp/pspropl/plthrp3 < dotmp3

 chmod 777 plot.ps0
mv plot.ps0 `echo 'finxxxvlcm.ps'|sed "s/xxx/$1/"`
lp `echo  'finxxxvlcm.ps' | sed "s/xxx/$1/"`

chmod 777 do*
```

Appendix 3 : Microstructure data processing
a.) script controlling all the microstructure processing

This shell program runs other shell programs, that each complete a component of the HRP microstructure data processing. The GUI tool allows the processing to be done all at once, as in this script or step by step. Each of the scripts called by this one is listed in the subsequent sections of appendix 3. The byte–swapped version of the raw data must exist before this script is executed.

```
#!/bin/sh
#
# bbtre2_allmic.sh
#
# shell script to run  the microstructure processing to get chi's and eps histograms  plotted
#            emontgomery 2/11/96
#
# the steps completed by this script are :  deconvolution of micro C & T (deconv)
#                                           Fourrier transforming the data  (mfft_bbtre2)
#                                           output diag. plts               (diagplt)
#                                           create summary files            (mall_bbtre2 )
#                                           format for plots                (mproc_sgi)
#                                           make plots                       (al_chieps)
#
# 7 command line arguments needed :         stn #
#                                           pressure bias
#                                           pr. at bottom of the mixed layer
#                                           pressure maximum
#                                           Sx shear probe sensitivity
#                                           Sy shear probe sensitivity
#                                           plot axis length
# ------------------------------------------------------------------------------
#
# check for presence of arguments, and exit if not given
 if [ $# -ne 7 ]
  then
    echo
    echo ' !!!!  you need to use 7 arguments !!!!'
    echo  ' please enter: stn #, pbias, bot of ML, pmax, sx sensitivity, sy sensitivity, and stick \
       plot  axis length'
    echo ' ===>  please use 3 characters for the station number'
    exit
 fi

# set the umask so the protections will allow group access
 umask 111

# now invoke the scripts that do each stage of the processing
 echo
 echo starting chi and eps computations with deconv      # Appendix 3.b
 echo ////////////////////////////////////////////////////

 sh $PROC_PATH/deconv.sh $1
 echo
 echo  now running mfft_g2                                          # Appendix 3.c
```

```
echo ////////////////////////////////////////////////////
sh $PROC_PATH/mfft_bbtre2.sh $1 $2 $3 $4 $5 $6

echo
echo now outputting diags plot to printer                    # Appendix 3.d
echo ////////////////////////////////////////////////////
sh $PROC_PATH/diagplt.sh $1

echo
echo doing mall_sgi to output the summary files              # Appendix 3.e
echo ////////////////////////////////////////////////////

sh $PROC_PATH/mall_bbtre2.sh $1 $4

#
#  we have to  do a little math here to get the last argument needed for the alchieps.sh, which is
# called by  mproc_sgi.sh.  With a vairable X-axis length, the number of plots to make has to be
# computed from the maximum pressure of the profile
#
 num=`expr $4 / $7`
 n=`expr $4 % $7`
 if [ $n -ne  0 ]
   then
     p=`expr $num + 1`
   else
     p=$num
 fi
 end=`expr $7 \* $p`

 echo
 echo 'running mproc to get the files needed to make the stickplots'
 echo 'and make .ps files (they do NOT automatically get printed)'
 echo ////////////////////////////////////////////////////

 sh $PROC_PATH/mproc_sgi.sh $1 $7 $end                       # Appendix 3.f

# end of script that automates the microstructure processing
```

Appendix 3 : Microstructure data processing
b.) script to run deconv

This script executes the deconvolution step of the micro processing. This is the first step in the microstructure processing. The template files required are decon_c.inp, and decon_t.inp.

```
#!/bin/sh
#
#  deconv.sh
#    script to generate the deconvolved c & t files to be used by mfft
#  MOD STB 3/13/97
#  copied needed templates into local dir
#  aded variables MICRO_PATH, FINE_PATH, and CRUISE

# check for presence of arguments, and exit if not given
if [ $# -eq 0 ]
then
 echo 'you need to use 1 argument: stn #'
 echo '   please enter 3 characters for the station number'
 exit
fi
#
#  set the umask so the protections will allow group access
umask 111

cd $WORK_PATH
# ==========================================================================
# remove the files created, if they already exist
rm `echo  $MICRO_PATH/$CRUISE'/hrp/deconv/deconv_b.cxxx' | sed "s/xxx/$1/"`
rm `echo  $MICRO_PATH/$CRUISE'/hrp/deconv/deconv_b.txxx' | sed "s/xxx/$1/"`

echo ' '
echo  ' running micro-c deconv now'
echo '============================='

sed "s/888/$1/g; s/fff/$FINN/g; s/mmm/$MICR/g; s/ccc/$CRUISE/g; s/lll/$CRUIS_LETTER/g" \
  $PROC_PATH/decon_c.inp > dodeconv
/sea1/prgs/hrp/deconv/deconv < dodeconv > dec_c

echo ' '
echo ' running micro-t deconv now'
echo '============================='

sed "s/888/$1/g; s/fff/$FINN/g; s/mmm/$MICR/g; s/ccc/$CRUISE/g; s/lll/$CRUIS_LETTER/g" \
  $PROC_PATH/decon_t.inp > dodeconv
/sea1/prgs/hrp/deconv/deconv < dodeconv > dec_t

chmod 777 $MICRO_PATH/$CRUISE/hrp/deconv/*
echo ' '
echo ' finished with deconv'
echo '  ***********************************'

# end of deconv script
```

Appendix 3 : Microstructure data processing
c.) script to run mfft_bbtre2

This shell script controls the execution of the program that computes the spectra. The data must have been deconvolved prior to execution. The template file required is mfft.inp.

```
#!/bin/sh
##  mfft_bbtre2.sh
#   script to  generate the mfft files
#
#  MOD 3/13/97  STB
#   add variables FINE_PATH, MICRO_PATH, and CRUISE
#   copied mfft.inp into proc directory
#  MOD 3/22/97 STB
#   add variables FINN, MICR, CRUIS_LETTER
# ***************************************************

# check for presence of arguments, and exit if not given
if [ $# -ne 6 ]
then
 echo  'you need to use 6 arguments: stn #, pbias,bot of ML, pmax, and the sx and sy calibrations'
 echo  '  please enter 3 characters for the station number'
 exit
fi
#  set the umask so the protections will allow group access
umask 111
# ====================================================================
# this is the second step of the micro processing, follows deconv
#
cd $WORK_PATH
sed "s/888/$1/g; s/777/$2/g; s/999/$3/g; s/890/$4/g;  s/892/$5/g; s/894/$6/g; s/fff/$FINN/g; \
    s/mmm/$MICR/g; s/ccc/$CRUISE/g; s/lll/$CRUIS_LETTER/g"  $PROC_PATH/mfft.inp \
    > domfft

# remove the files created, if they already exist
rm 'echo $MICRO_PATH/$CRUISE'/hrp/mfft/mfft_g2.dxxx' | sed "s/xxx/$1/"'
rm 'echo $MICRO_PATH/$CRUISE'/hrp/mfft/diags_g2.dxxx' | sed "s/xxx/$1/"'
rm  'echo  $MICRO_PATH/$CRUISE'/hrp/mfft/mrun_g2.dxxx' | sed "s/xxx/$1/"'
rm mrun.log

echo ' '
echo ' running mfft now, output is directed to mrun.log'
echo ' this could take awhile, so  be patient....'
echo '=========================================================='
/sea1/prgs/hrp/mfft/mfft_bbtre2 < domfft > mrun.log

chmod 777 $MICRO_PATH/$CRUISE/hrp/mfft/*
#
# now start putting together the processing log: store it in the same  directory as the data
cat domfft 'echo $MICRO_PATH/$CRUISE'/hrp/mfft/mrun_g2.dxxx' | sed "s/xxx/$1/"' > \
  'echo $MICRO_PATH/$CRUISE'/hrp/mfft/mrunxxx.log' |  sed "s/xxx/$1/"'

echo ' '
echo '   mfft_bbtre2.sh finished'
echo ' '
echo '  ********************************************'
```

Appendix 3 : Microstructure data processing
d.) script to make diagnostics plot

PlotPlus software creates the diagnostics plot, of variables generated by the execution of mfft. The template file used, diags.ppc, is listed after the script as an example of a PlotPlus input file.

```
#!/bin/sh
# diagplt.sh
#   generates the mfft diagnostic plot
#
# check for presence of arguments, and exit if not given
if [ $# -ne 1 ]
then
  echo 'you need to use a command line arguement: station number'
  echo '    please enter 3 characters for the number– ie. use 001 not 1'
  exit
fi
#  set the umask so the protections will allow group access
umask 111

cd $WORK_PATH
rm mpl0*
echo  doing the mfft diags plot now.
sed "s/888/$1/g; s/mmm/$MICR/g; s/ccc/$CRUISE/g" $PROC_PATH/diags.ppc  > diagplt
  /usr/local/bin/pplus diagplt

m2ps mpl001 > `echo $WORK_PATH/diag"$1".ps"`
lp `echo $WORK_PATH/diag"$1".ps"`
chmod 777 *
# ==========end of diagnostic plot generation ============================
```

diags.ppc:

```
format, unf
size 10.5,8.0
axlen 7,5.5
origin 2.2,1.8
xlab Pressure
pltype 0
window on
box off
xaxis 0.0,5000.,1000.
yaxis –.004,0.0,0.0001
axlint,1,5
xfor,(I4)
yfor,(f8.5)
line 1,1,0
c vars 1,1,2,26
vars,1,1,2,,,,,,,,,,,,,,,,,,,,,,
rd, /mmm/data/ccc/hrp/mfft/diags_g2.d888
line 2,1,0
c vars 1,1,,2,26
vars,1,1,,2,,,,,,,,,,,,,,,,,,,,,
rd, /mmm/data/ccc/hrp/mfft/diags_g2.d888
c list data
plot bbtre2 dive 888 25 – 4500.0 dB
exit
```

Appendix 3 : Microstructure data processing
e.) script to run mall_bbtre2

This script organizes input and executes the program that creates the ascii summary files which are the source of the chi and epsilon estimates from which quanties like diffusivity are computed. The template file required is PROC_template.DAT, which after sed becomes PROC.DAT.

*** The program is hardwired to use a file called PROC.DAT, so do NOT modify the name used. Anywhere else in the system, names don't matter, but PROC.DAT has to remain PROC.DAT

```
#!/bin/sh
#
# mall_bbtre2.sh
#    script to generate the ascii summary files that contain the epsilon and chi estimates
#
# MOD STB 3/13/97
#   add variables FINE_PATH, MICRO_PATH, and CRUISE
#   put template file into proc directory and renamed to
#   PROC_template.DAT so it does not clobber the input to mall_sgi

# check for arguments, and exit if not given
if [ $# -ne 2 ]
then
 echo 'you need two arguments: station number, and max. pressure'
 echo '   please enter 3 characters for stn. # – i.e. use 001 not 1'
 exit
fi
# set the umask so the protections will allow group access
umask 111

cd $WORK_PATH
# ==========================================================================
# set up the PROC.DAT file needed for mall to run

echo ' setting up PROC.DAT file now'
sed "s/888/$1/g; s/999/$2/g; s/mmm/$MICR/g; s/ccc/$CRUISE/g" \
    $PROC_PATH/PROC_template.DAT > PROC.DAT

echo ' running mall'

/sea1/prgs/hrp/mall/mall_bbtre2

chmod 777 $MICRO_PATH/$CRUISE/hrp/summ/*.d*

#
# append proc.dat to the processing log–  $MICRO_PATH/$CRUISE/hrp/mfft/mrunxxx.log
#
cat `echo $MICRO_PATH/$CRUISE'/hrp/mfft/mrunxxx.log' | sed "s/xxx/$1/"` PROC.DAT > \
  `echo  $MICRO_PATH/$CRUISE'/hrp/summ/mrunxxx.log' | sed "s/xxx/$1/"`

echo ' '
echo '  finished with mall.sh'
echo ' ****************************************'
echo '
```

Appendix 3 : Microstructure data processing.
f.) script to reformat chi and eps for histogram plotting

This script reformats the data in the summary files created by mall to make histogram plots. The template file required for this script is do_mp_sgi. Mproc_sqi.sh calls another script, alchieps.sh to handle the actual plot generation using PlotPlus. Alchieps.sh is listed in the next appendix.

```
#!/bin/sh
#
#  mproc_sgi.sh
#   shell script to use the template and run mproc to reformat summary data for plotting filtering
#   with thedata quality words for each scan
#
# MOD STB 3/13/97
#    added variables FINE_PATH and CRUISE  & copied template file into  proc directory
#*****************************

# check for presence of arguements, and exit if not given
if [ $# -ne 3 ]
then
 echo  'you need to use 3 command line arguments: stn#, axis length'
 echo  'for the chi and eps plots, and pmax, which must be a multiple'
 scho  'of the axis length'
 echo  '    please enter 3 characters for the stn # – i.e. use 001 not 1'
 exit
fi
#
# set the umask so the protections will allow group access
umask 111

# ==========================================================================
echo doing the mproc run now.
echo   currently reading SGI generated data
#
 cd $WORK_PATH
 sed "s/888/$1/g; s/mmm/$MICR/g; s/ccc/$CRUISE/g" $PROC_PATH/do_mp_sgi > domproc1

 /sea1/prgs/hrp/mproc/mproc_sgi < domproc1


# now invoke alchieps to make the plots....
sh $PROC_PATH/alchieps.sh $1 $2 $3

# end of mproc_sgi
```

Appendix 3 : Microstructure data processing
g.) script to generate histogram plots

This script provides a versatile method of generating the chi–c, chi–t and epsilon histogram plots that are the main output of the microstructure data processing.

Normally, the plots are made displaying 600 meters/page , so a 4800 meter profile of epsilon would be broken into 8 pages: 0–600, 600–1200, 1200–1800, 1800–2400, 2400–3000, 3000–3600, 3600–4200, and 4200–4800. The same axis scaling is used for chi–c, chi–t and epsilon, so, if 600 m increments were chosen, 24 histogram plots will be made. This script allows the whole profile to be plotted on one page or many, depending on the input.

```
#!/bin/sh
#
#  alchieps.sh
#     shell script to automate the edits needed to make all the eps and chi plots
#   emontgomery 1/30/96
#
# MOD STB 3/13/97
#   add variables FINE_PATH, MICRO_PATH, and CRUISE
#   put template files into proc directory
# ******************************************************

# check for presence of arguments, and exit if not given
#   3 arguments are needed  station#, pressure_interval_to_plot, & pmax
if [ $# –ne 3 ]
then
  echo 'you need to use 3 arguments: stn #, pressure_interval_to_plot'
  echo '  and pmax'
  echo '  please enter 3 characters for the station number'
  echo '  600 is a normal pressure interval to  use, and pmax should'
  echo '  be an even multiple of the pressure interval'
  exit
fi
#
# set the umask so the protections will allow group access
umask 111

cd $WORK_PATH
rm mpl0*

# assign some values to variables to use
 win=$2
  pmax=$3
  winstart=0
  sta=$1
  knt=1
  echo  for Station $sta

# this loops through once each for eps, chi–t, and chi–c at each depth range and makes the plot

 while [ $winstart –lt $pmax ]
  do
    winend=‘expr $winstart + $win‘
#     skp=‘expr $knt \* 3000‘
```

```
   echo $skp
   echo
   echo ==========================================
   echo ' the window is now : ' $winstart $winend

   echo  ' and the pressure maximum is' $pmax
   echo ==========================================

#   make the eps ppc file and plot it for cycle x
    echo    doing EPS
    sed "s/XXX/$sta/g; s/aaa/$winstart/g; s/bbb/$winend/g; s/mmm/$MICR/g; \
       s/crus/$CRUISE/g"  $PROC_PATH/ep_sgi_tmpl > eps.ppc
    /usr/local/bin/pplus eps.ppc
    /usr/local/bin/m2ps mpl001 > 'echo 'epsx.ps' | sed "s/x/$knt/"'
    rm mpl0*


#   make the chit ppc file and plot it for cycle x
    echo    doing CHIT
    sed "s/XXX/$sta/g; s/aaa/$winstart/g; s/bbb/$winend/g;  s/mmm/$MICR/g; \
       s/crus/$CRUISE/g"  $PROC_PATH/ct_sgi_tmpl > chit.ppc
    /usr/local/bin/pplus chit.ppc
    /usr/local/bin/m2ps mpl001 > 'echo 'chitx.ps' | sed "s/x/$knt/"'
     rm mpl0*


#   make the chic ppc file and plot it for cycle x
    echo doing CHIC
    sed "s/XXX/$sta/g; s/aaa/$winstart/g; s/bbb/$winend/g; s/mmm/$MICR/g;  \
       s/crus/$CRUISE/g"  $PROC_PATH/cc_sgi_tmpl > chic.ppc
    /usr/local/bin/pplus chic.ppc
    /usr/local/bin/m02ps mpl001 > 'echo 'chicx.ps' | sed "s/x/$knt/"'
     rm mpl0*

#   increment pmin for window and counter
    winstart='echo $winend'
    knt='expr $knt + 1'
#     echo $knt
 done

# now print the .ps files and then delete them
#  This is currently done separately from the GUI tool, but could be put back in here
#  if desired
#   lp eps*ps
#   lp chit*ps
#   lp chic*ps
#   rm eps*ps chi*ps

# end of plot generation script
```

DOCUMENT LIBRARY

Distribution List for Technical Report Exchange – September 1997

University of California, San Diego
SIO Library 0175C
9500 Gilman Drive
La Jolla, CA 92093-0175

Hancock Library of Biology & Oceanography
Alan Hancock Laboratory
University of Southern California
University Park
Los Angeles, CA 90089-0371

Gifts & Exchanges
Library
Bedford Institute of Oceanography
P.O. Box 1006
Dartmouth, NS, B2Y 4A2, CANADA

NOAA/EDIS Miami Library Center
4301 Rickenbacker Causeway
Miami, FL 33149

Research Library
U.S. Army Corps of Engineers
Waterways Experiment Station
3909 Halls Ferry Road
Vicksburg, MS 39180-6199

Institute of Geophysics
University of Hawaii
Library Room 252
2525 Correa Road
Honolulu, HI 96822

Marine Resources Information Center
Building E38-320
MIT
Cambridge, MA 02139

Library
Lamont-Doherty Geological Observatory
Columbia University
Palisades, NY z10964

Library
Serials Department
Oregon State University
Corvallis, OR 97331

Pell Marine Science Library
University of Rhode Island
Narragansett Bay Campus
Narragansett, RI 02882

Working Collection
Texas A&M University
Dept. of Oceanography
College Station, TX 77843

Fisheries-Oceanography Library
151 Oceanography Teaching Bldg.
University of Washington
Seattle, WA 98195

Library
R.S.M.A.S.
University of Miami
4600 Rickenbacker Causeway
Miami, FL 33149

Maury Oceanographic Library
Naval Oceanographic Office
Building 1003 South
1002 Balch Blvd.
Stennis Space Center, MS, 39522-5001

Library
Institute of Ocean Sciences
P.O. Box 6000
Sidney, B.C. V8L 4B2
CANADA

National Oceanographic Library
Southampton Oceanography Centre
European Way
Southampton SO14 3ZH
UK

The Librarian
CSIRO Marine Laboratories
G.P.O. Box 1538
Hobart, Tasmania
AUSTRALIA 7001

Library
Proudman Oceanographic Laboratory
Bidston Observatory
Birkenhead
Merseyside L43 7 RA
UNITED KINGDOM

IFREMER
Centre de Brest
Service Documentation - Publications
BP 70 29280 PLOUZANE
FRANCE

50272-101

REPORT DOCUMENTATION PAGE	1. REPORT NO. WHOI-98-04	2.	3. Recipient's Accession No.

4. Title and Subtitle
A Graphical User Interface for Processing Data from the High Resolution Profiler (HRP)

5. Report Date
March 1998

6.

7. Author(s) Ellyn T. Montgomery and S. Thompson Bolmer

8. Performing Organization Rept. No.
WHOI-98-04

9. Performing Organization Name and Address

Woods Hole Oceanographic Institution
Woods Hole, Massachusetts 02543

10. Project/Task/Work Unit No.

11. Contract(C) or Grant(G) No.
(C) OCE-94-15589
(G)

12. Sponsoring Organization Name and Address

National Science Foundation

13. Type of Report & Period Covered
Technical Report

14.

15. Supplementary Notes
This report should be cited as: Woods Hole Oceanog. Inst. Tech. Rept., WHOI-98-04.

16. Abstract (Limit: 200 words)

The High Resolution Profiler (HRP) is one of the only oceanographic instruments that is capable of measuring turbulent velocity and temperature fluctuations in the abyssal ocean. It is a unique device, and consequently specialized communications, data conversion and analysis software are employed to examine the data it collects.

This document describes a major upgrade of the software and hardware systems used to process data from the HRP. The bulk of the conversion occurred in 1996 prior to the Brazil Basin Tracer Release Experiment (BBTRE). During the upgrade process, a Graphical User Interface (GUI) was designed and implemented for accomplishing routine HRP data processing tasks.

17. Document Analysis a. Descriptors
High Resolution Profiler
software upgrade
data manipulation

b. Identifiers/Open-Ended Terms

c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
Approved for public release; distribution unlimited.	UNCLASSIFIED	45
	20. Security Class (This Page)	22. Price

(See ANSI-Z39.18) *See Instructions on Reverse*

OPTIONAL FORM 272 (4-77)
(Formerly NTIS-35)
Department of Commerce